Package 'Ckmeans.1d.dp'

September 7, 2019

Type Package

Title Optimal, Fast, and Reproducible Univariate Clustering

Version 4.3.0

Date 2019-09-06

Author Joe Song [aut, cre],
Hua Zhong [aut],
Haizhou Wang [aut]

Maintainer Joe Song <joemsong@cs.nmsu.edu>

Description Fast, optimal, and reproducible weighted univariate clustering by dynamic programming. Four types of problem including univariate k-means, k-median, k-segments, and multi-channel weighted k-means are solved with guaranteed optimality and reproducibility. The core algorithm minimizes the sum of (weighted) within-cluster distances using respective metrics. Its advantage over heuristic clustering in efficiency and accuracy is pronounced at a large number of clusters k. Weighted k-means can also process time series to perform peak calling. Multi-channel weighted k-means groups multiple univariate signals into k clusters. An auxiliary function generates histograms that are adaptive to patterns in data. This package provides a powerful set of tools for univariate data analysis with guaranteed optimality, efficiency, and reproducibility.

License LGPL (>= 3)

Encoding UTF-8

LazyData true

Imports Rcpp, Rdpack

LinkingTo Rcpp

NeedsCompilation yes

Suggests testthat, knitr, rmarkdown

RdMacros Rdpack

VignetteBuilder knitr

Repository CRAN

Date/Publication 2019-09-07 11:20:07 UTC

R topics documented:

Ckmeans.1d.dp-package . . . 2
ahist . . . 4
MultiChannel.WUC . . . 7
plot.Ckmeans.1d.dp . . . 9
plot.Cksegs.1d.dp . . . 10
plotBIC . . . 11
print.Ckmeans.1d.dp . . . 12
print.Cksegs.1d.dp . . . 13
Univariate Clustering . . . 14
Univariate Segmentation . . . 18

Index **21**

`Ckmeans.1d.dp-package` *Optimal, Fast, and Reproducible Univariate Clustering*

Description

This package provides fast optimal univariate clustering by dynamic programming. Four types of problem including univariate k-means, k-median, k-segments, and multi-channel weighted k-means are solved with guaranteed optimality and reproducibility. The core algorithm minimizes the (weighted) sum of within-cluster distances using respective metrics. Its advantage over heuristic clustering in efficiency and accuracy is increasingly pronounced as the number of clusters k increases. Weighted k-means can also optimally segment time series to perform peak calling. An auxiliary function generates histograms that are adaptive to patterns in data. This package provides a powerful set of tools for univariate data analysis with guaranteed optimality, efficiency, and reproducibility.

The Ckmeans.1d.dp algorithm clusters (weighted) univariate data given by a numeric vector x into k groups by dynamic programming (Wang and Song 2011). It guarantees the optimality of clustering—the total of within-cluster sums of squares is always the minimum given the number of clusters k. In contrast, heuristic univariate clustering algorithms may be non-optimal or inconsistent from run to run. As unequal non-negative weights are supported for each point, the algorithm can also segment a time course using the time points as input and the values at each time point as weight. Utilizing the optimal clusters, a function can generate histograms adaptive to patterns in data. This method is numerically stable.

Apart from the time for sorting x, the weighted univariate clustering algorithm takes a runtime of $O(kn)$, linear in both sample size n and the number of clusters k. The new approach was first introduced into version 3.4.9 (not publicly released) on July 16, 2016, using a new divide-and-conquer strategy integrating a previous theoretical result on matrix search (Aggarwal et al. 1987) and a novel in-place search space reduction method. Earlier versions since 3.4.6 used a divide-and-conquer strategy in dynamic programming to reduce the runtime from $O(kn^2)$ down to $O(kn \lg n)$.

Bellman (1973) first described a general dynamic programming strategy for solving univariate clustering problems with additive optimality measures. The strategy, however, did not address any specific characteristics of the k-means problem and its implied general algorithm will have a time complexity of $O(kn^3)$ on an input of n points.

The space complexity in all cases is $O(kn)$.

This package provides a powerful alternative to heuristic clustering and also new functionality for weighted clustering, segmentation, and peak calling with guaranteed optimality. It is practical for Ckmeans.1d.dp to cluster millions of sample points with optional weights in seconds using a single processor on a not very recent desktop computer.

Third parties have ported various versions of this package to JavaScript, MATLAB, Python, Ruby, SAS, and Scala.

Details

Package:	Ckmeans.1d.dp
Type:	Package
Version:	4.3.0
Initial release version:	1.0
Initial release date:	2010-10-26
License:	LGPL (>= 3)

The most important function of this package is `Ckmeans.1d.dp` that implements optimal univariate clustering either weighted or unweighted. It also contains an adaptive histogram function `ahist`, plotting functions `plot.Ckmeans.1d.dp` and `plotBIC`, and a print function `print.Ckmeans.1d.dp`.

Disclaimer

Author(s)

Joe Song, Hua Zhong, and Haizhou Wang

References

Aggarwal A, Klawe MM, Moran S, Shor P, Wilber R (1987). "Geometric applications of a matrix-searching algorithm." *Algorithmica*, **2**(1-4), 195–208.

Bellman R (1973). "A note on cluster analysis and dynamic programming." *Mathematical Biosciences*, **18**(3), 311–312.

Wang H, Song M (2011). "Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming." *The R Journal*, **3**(2), 29–33. https://doi.org/10.32614/RJ-2011-015.

See Also

The kmeans function in package **stats** that implements several heuristic k-means algorithms.

---

ahist *Adaptive Histograms*

Description

Generate or plot histograms adaptive to patterns in univariate data. The number and widths of histogram bins are automatically calculated based on an optimal univariate clustering of input data. Thus the bins are unlikely of equal width.

Usage

```
ahist(x, k = c(1,9), breaks=NULL, data=NULL, weight=1,
      plot = TRUE, xlab = deparse(substitute(x)),
      wlab = deparse(substitute(weight)),
      main = NULL, col = "lavender", border = graphics::par("fg"),
      lwd = graphics::par("lwd"),
      col.stick = "gray", lwd.stick = 1, add.sticks=TRUE,
      style = c("discontinuous", "midpoints"),
      skip.empty.bin.color=TRUE,
      ...)
```

Arguments

x	a numeric vector of data or an object of class "Ckmeans.1d.dp". If x is a numeric vector, all NA elements must be removed from x before calling this function. If x is an object of class "Ckmeans.1d.dp", the clustering information in x will be used and the data argument contains the numeric vector to be plotted.
k	either an exact integer number of bins/clusters, or a vector of length two specifying the minimum and maximum numbers of bins/clusters to be examined. The default is c(1,9). When k is a range, the actual number of clusters is determined by Bayesian information criterion. This argument is ignored if x is an object of class "Ckmeans.1d.dp".
breaks	This argument is defined in hist. If this argument is provided, optimal univariate clustering is not applied to obtain the histogram, but instead the histogram will be generated by the hist function in **graphics**, except that sticks representing data can still be optionally plotted by specifying the add.sticks=TRUE argument.
data	a numeric vector. If x is an object of class "Ckmeans.1d.dp", the data argument must be provided. If x is a numeric vector, this argument is ignored.

weight	a value of 1 to specify equal weights or a numeric vector of unequal weights for each element. The default weight is one. It is highly recommended to use positive (instead of zero) weights to account for the influence of every element. The weights have a strong impact on the clustering result.
plot	a logical. If TRUE, the histogram will be plotted.
xlab	a character string. The x-axis label for the plot.
wlab	a character string. The weight-axis label for the plot. It is the vertical axis to the right of the plot.
main	a character string. The title for the plot.
col	a character string. The fill color of the histogram bars.
border	a character string. The color of the histogram bar borders.
lwd	a numeric value. The line width of the border of the histogram bars
col.stick	a character string. The color of the sticks above the x-axis. See Details.
lwd.stick	a numeric value. The line width of the sticks above the x-axis. See Details.
add.sticks	a logical. If TRUE (default), the sticks representing the data will be added to the bottom of the histogram. Otherwise, sticks are not plotted.
style	a character string. The style of the adaptive histogram. See details.
skip.empty.bin.color	a logical. If TRUE (default), an empty bin (invisible) will be assigned the same bar color with the next bin. This is useful when all provided colors are to be used for non-empty bins. If FALSE, each bin will be assigned a bar color from col. A value of TRUE will coordinate the bar and stick colors.
...	additional arguments to be passed to hist or plot.histogram.

Details

The histogram is by default plotted using the plot.histogram method. The plot can be optionally disabled with the plot=FALSE argument. The original input data are shown as sticks just above the horizontal axis.

If the breaks argument is not specified, the number of histogram bins is the optimal number of clusters estimated using Bayesian information criterion evaluated on Gaussian mixture models fitted to the input data in x.

If not provided with the breaks argument, breaks in the histogram are derived from clusters identified by optimal univariate clustering (Ckmeans.1d.dp) in two styles. With the default "discontinuous" style, the bin width of each bar is determined according to a data-adaptive rule; the "midpoints" style uses the midpoints of cluster border points to determine the bin-width. For clustered data, the "midpoints" style generates bins that are too wide to capture the cluster patterns. In contrast, the "discontinuous" style is more adaptive to the data by allowing some bins to be empty making the histogram bars discontinuous.

Value

An object of class histogram defined in hist. It has a S3 plot method plot.histogram.

Author(s)

Joe Song

See Also

hist in package **graphics**.

Examples

```
# Example 1: plot an adaptive histogram from data generated by
#   a Gaussian mixture model with three components
x <- c(rnorm(40, mean=-2, sd=0.3),
       rnorm(45, mean=1, sd=0.1),
       rnorm(70, mean=3, sd=0.2))
ahist(x, col="lightblue", sub=paste("n =", length(x)),
      col.stick="salmon", lwd=2,
      main=paste("Example 1. Gaussian mixture model with 3 components",
                 "(one bin per component)", sep="\n"))


# Example 2: plot an adaptive histogram from data generated by
#   a Gaussian mixture model with three components using a given
#   number of bins
ahist(x, k=9, col="lavender", col.stick="salmon",
      sub=paste("n =", length(x)), lwd=2,
      main=paste("Example 2. Gaussian mixture model with 3 components",
                 "(on average 3 bins per component)", sep="\n"))

# Example 3: The DNase data frame has 176 rows and 3 columns of
#   data obtained during development of an ELISA assay for the
#   recombinant protein DNase in rat serum.

data(DNase)
res <- Ckmeans.1d.dp(DNase$density)
kopt <- length(res$size)
ahist(res, data=DNase$density, col=rainbow(kopt),
      col.stick=rainbow(kopt)[res$cluster],
      sub=paste("n =", length(x)), border="transparent",
      xlab="Optical density of protein DNase",
      main="Example 3. Elisa assay of DNase in rat serum")


# Example 4: Add sticks to histograms with the R provided
#   hist() function.

ahist(DNase$density, breaks="Sturges", col="palegreen",
      add.sticks=TRUE, col.stick="darkgreen",
      main=paste("Example 4. Elisa assay of DNase in rat serum",
                 "(Equal width bins)", sep="\n"),
      xlab="Optical density of protein DNase")

# Example 5: Weighted adatpive histograms
```

```
x <- sort(c(rnorm(40, mean=-2, sd=0.3),
       rnorm(45, mean=2, sd=0.1),
       rnorm(70, mean=4, sd=0.2)))

y <- (1 + sin(0.10 * seq_along(x))) * (x-1)^2

ahist(x, weight=y, sub=paste("n =", length(x)),
      col.stick="forestgreen", lwd.stick=0.25, lwd=2,
      main="Example 5. Weighted adaptive histogram")


# Example 6: Cluster data with repetitive elements

ahist(c(1,1,1,1, 3,4,4, 6,6,6), k=c(2,4), col="gray",
      lwd=2, lwd.stick=6, col.stick="chocolate",
      main=paste("Example 6. Adaptive histogram of",
                 "repetitive elements", sep="\n"))
```

`MultiChannel.WUC` *Optimal Multi-channel Weighted Univariate Clustering*

Description

Perform optimal multi-channel weighted univariate k-means clustering in linear time.

Usage

```
MultiChannel.WUC(x, y,  k=c(1,9))
```

Arguments

`x`	a numeric vector of data to be clustered. All `NA` elements must be removed from `x` before calling this function. The function will run faster on sorted `x` (in non-decreasing order) than an unsorted input.
`y`	a numeric matrix of non-negative weights for each element in `x`. Columns of the matrix are channels. It is highly recommended to use positive (instead of zero) weights to account for the influence of every element. Weights strongly influence clustering results. When the number of clusters `k` is given as a range, the weights should be linearly scaled to sum up to the observed sample size.
`k`	either an exact integer number of clusters, or a vector of length two specifying the minimum and maximum numbers of clusters to be examined. The default is `c(1,9)`. When `k` is a range, the actual number of clusters is determined by Bayesian information criterion (BIC).

Details

`MultiChannel.WUC` minimizes the total weighted within-cluster sum of squared distance. It uses the SMAWK algorithm (Aggarwal et al. 1987) with modified data structure to speed up the dynamic programming to linear runtime. The method selects an optimal `k` based on an approximate Gaussian mixture model using the BIC.

Value

A list object containing the following components:

`cluster`	a vector of clusters assigned to each element in `x`. Each cluster is indexed by an integer from 1 to `k`.
`centers`	a numeric vector of the (weighted) means for each cluster.
`withinss`	a numeric vector of the (weighted) within-cluster sum of squares for each cluster.
`size`	a vector of the (weighted) number of elements in each cluster.
`totss`	total sum of (weighted) squared distances between each element and the sample mean. This statistic is not dependent on the clustering result.
`tot.withinss`	total sum of (weighted) within-cluster squared distances between each element and its cluster mean. This statistic is minimized given the number of clusters.
`betweenss`	sum of (weighted) squared distances between each cluster mean and sample mean. This statistic is maximized given the number of clusters.
`xname`	a character string. The actual name of the `x` argument.
`yname`	a character string. The actual name of the `y` argument.

Author(s)

Hua Zhong and Mingzhou Song

References

Aggarwal A, Klawe MM, Moran S, Shor P, Wilber R (1987). "Geometric applications of a matrix-searching algorithm." *Algorithmica*, **2**(1-4), 195–208.

Examples

```
X <- sort(sample(x = c(1:100), size = 20, replace = TRUE))
Y <- matrix(sample(x = c(1:100), size = 40, replace = TRUE), ncol=2, nrow=length(X))

res <- MultiChannel.WUC(x = X, y = Y, k = c(5:10))
```

`plot.Ckmeans.1d.dp` *Plot Optimal Univariate Clustering Results*

Description

Plot optimal univariate clustering results returned from `Ckmeans.1d.dp`.

Usage

```
## S3 method for class 'Ckmeans.1d.dp'
plot(x, xlab=NULL, ylab=NULL, main=NULL,
      sub=NULL, col.clusters=NULL, ...)
## S3 method for class 'Ckmedian.1d.dp'
plot(x, xlab=NULL, ylab=NULL, main=NULL,
      sub=NULL, col.clusters=NULL, ...)
```

Arguments

`x`	an object of class as returned by `Ckmeans.1d.dp` or `Ckmedian.1d.dp`.
`xlab`	a character string. The x-axis label for the plot.
`ylab`	a character string. The x-axis label for the plot.
`main`	a character string. The title for the plot.
`sub`	a character string. The subtitle for the plot.
`col.clusters`	a vector of colors, defined either by integers or by color names. If the length is shorter than the number of clusters, the colors will be reused.
`...`	arguments passed to `plot` function in package **graphics**.

Details

The functions `plot.Ckmeans.1d.dp` and `plot.Ckmedian.1d.dp` visualize the input data as sticks whose heights are the weights. They use different colors to indicate clusters.

Value

An object of class "`Ckmeans.1d.dp`" or "`Ckmedian.1d.dp`" defined in `Ckmeans.1d.dp`.

Author(s)

Joe Song

Examples

```
# Example: clustering data generated from a Gaussian
#          mixture model of three components
x <- c(rnorm(50, mean=-1, sd=0.3),
       rnorm(50, mean=1, sd=0.3),
       rnorm(50, mean=3, sd=0.3))

res <- Ckmeans.1d.dp(x)
plot(res)

y <- (rnorm(length(x)))^2
res <- Ckmeans.1d.dp(x, y=y)
plot(res)

res <- Ckmedian.1d.dp(x)
plot(res)
```

`plot.Cksegs.1d.dp` *Plot Optimal Univariate Segmentation Results*

Description

Plot optimal univariate segmentation results returned from `Cksegs.1d.dp`.

Usage

```
## S3 method for class 'Cksegs.1d.dp'
plot(x, xlab=NULL, ylab=NULL, main=NULL,
      sub=NULL, col.clusters=NULL, ...)
```

Arguments

`x`	an object of class as returned by `Cksegs.1d.dp`.
`xlab`	a character string. The x-axis label for the plot.
`ylab`	a character string. The x-axis label for the plot.
`main`	a character string. The title for the plot.
`sub`	a character string. The subtitle for the plot.
`col.clusters`	a vector of colors, defined either by integers or by color names. If the length is shorter than the number of clusters, the colors will be reused.
`...`	arguments passed to `plot` function in package **graphics**.

Details

The function `plot.Cksegs.1d.dp` shows segments as horizontal lines from the univariate segmentation results obtained from function `Cksegs.1d.dp`. It uses different colors to indicate segments.

Value

An object of class "Cksegs.1d.dp" defined in Cksegs.1d.dp.

Author(s)

Joe Song

References

Wang, H. and Song, M. (2011) Ckmeans.1d.dp: optimal *k*-means clustering in one dimension by dynamic programming. *The R Journal* **3**(2), 29–33. Retrieved from https://journal.r-project.org/archive/2011-2/RJournal_2011-2_Wang+Song.pdf

Examples

```
# Example: clustering data generated from a Gaussian
#          mixture model of three components
x <- c(rnorm(50, mean=-1, sd=0.3),
       rnorm(50, mean=1, sd=0.3),
       rnorm(50, mean=3, sd=0.3))

y <- x^3
res <- Cksegs.1d.dp(y, x=x)
plot(res, lwd=2)
```

---

plotBIC *Plot Bayesian Information Criterion as a Function of Number of Clusters*

Description

Plot Bayesian information criterion (BIC) as a function of the number of clusters obtained from optimal univariate clustering results returned from Ckmeans.1d.dp. The BIC normalized by sample size (BIC/n) is shown.

Usage

```
plotBIC(
  ck, xlab="Number of clusters k",
  ylab = "BIC/n", type="b",
  sub=paste("n =", length(ck$cluster)),
  main=paste("Bayesian information criterion",
             "(normalized by sample size)", sep="\n"),
  ...
)
```

Arguments

ck	an object of class Ckmeans.1d.dp returned by Ckmeans.1d.dp.
xlab	a character string. The x-axis label for the plot.
ylab	a character string. The x-axis label for the plot.
type	the type of plot to be drawn. See plot.
main	a character string. The title for the plot.
sub	a character string. The subtitle for the plot.
...	arguments passed to plot function in package **graphics**.

Details

The function visualizes the input data as sticks whose heights are the weights. It uses different colors to indicate optimal k-means clusters.

Value

An object of class "Ckmeans.1d.dp" defined in Ckmeans.1d.dp.

Author(s)

Joe Song

Examples

```
# Example: clustering data generated from a Gaussian mixture
#          model of two components
x <- rnorm(50, mean=-1, sd=0.3)
x <- append(x, rnorm(50, mean=1, sd=0.3) )
res <- Ckmeans.1d.dp(x)
plotBIC(res)

y <- (rnorm(length(x)))^2
res <- Ckmeans.1d.dp(x, y=y)
plotBIC(res)
```

---

print.Ckmeans.1d.dp *Print Optimal Univariate Clustering Results*

Description

Print optimal univariate clustering results obtained from Ckmeans.1d.dp or Ckmedian.1d.dp.

Usage

```
## S3 method for class 'Ckmeans.1d.dp'
print(x, ...)
## S3 method for class 'Ckmedian.1d.dp'
print(x, ...)
```

Arguments

x	object returned by calling Ckmeans.1d.dp or Cksegs.1d.dp.
...	arguments passed to print function.

Details

Function print.Ckmeans.1d.dp and print.Ckmedian.1d.dp prints the maximum ratio of between-cluster sum of squares to total sum of squares unless all input elements are zero. The ratio is an indicator of maximum achievable clustering quality given the number of clusters: 100% for a perfect clustering and 0% for no cluster patterns.

Value

An object of class "Ckmeans.1d.dp" or "Ckmedian.1d.dp" as defined in Ckmeans.1d.dp.

Author(s)

Joe Song and Haizhou Wang

Examples

```
# Example: clustering data generated from a Gaussian
#          mixture model of two components
x <- c(rnorm(15, mean=-1, sd=0.3),
       rnorm(15, mean=1, sd=0.3))
res <- Ckmeans.1d.dp(x)
print(res)

res <- Ckmedian.1d.dp(x)
print(res)

y <- (rnorm(length(x)))^2
res <- Ckmeans.1d.dp(x, y=y)
print(res)

res <- Ckmedian.1d.dp(x)
print(res)
```

---

print.Cksegs.1d.dp *Print Optimal Univariate Segmentation Results*

Description

Print optimal univariate segmentation results obtained from Cksegs.1d.dp.

Usage

```
## S3 method for class 'Cksegs.1d.dp'
print(x, ...)
```

Arguments

x	object returned by calling Cksegs.1d.dp.
...	arguments passed to print function.

Details

Function print.Cksegs.1d.dp prints the maximum ratio of between-cluster sum of squares to total sum of squares unless all input elements are zero. The ratio is an indicator of maximum achievable clustering quality given the number of clusters: 100% for a perfect clustering and 0% for no cluster patterns.

Value

An object of class "Cksegs.1d.dp" as defined in Cksegs.1d.dp.

Author(s)

Joe Song

Examples

```
# Example: clustering data generated from a Gaussian
#          mixture model of two components
x <- c(rnorm(15, mean=-1, sd=0.3),
       rnorm(15, mean=1, sd=0.3))

y <- x^3
res <- Cksegs.1d.dp(y, x=x)
print(res, lwd=2)
```

Univariate Clustering *Optimal (Weighted) Univariate Clustering*

Description

Perform optimal univariate k-means or k-median clustering in linear (fastest), loglinear, or quadratic (slowest) time.

Usage

```
Ckmeans.1d.dp(x, k=c(1,9), y=1,
              method=c("linear", "loglinear", "quadratic"),
              estimate.k=c("BIC", "BIC 3.4.12"))

Ckmedian.1d.dp(x, k=c(1,9), y=1,
               method=c("linear", "loglinear", "quadratic"),
               estimate.k=c("BIC", "BIC 3.4.12"))
```

Arguments

`x`	a numeric vector of data to be clustered. All `NA` elements must be removed from `x` before calling this function. The function will run faster on sorted `x` (in non-decreasing order) than an unsorted input.
`k`	either an exact integer number of clusters, or a vector of length two specifying the minimum and maximum numbers of clusters to be examined. The default is `c(1,9)`. When `k` is a range, the actual number of clusters is determined by Bayesian information criterion.
`y`	a value of 1 (default) to specify equal weights of 1 for each element in `x`, or a numeric vector of unequal non-negative weights for each element in `x`. It is highly recommended to use positive (instead of zero) weights to account for the influence of every element. The weights have a strong impact on the clustering result. When the number of clusters `k` is given as a range, the weights should be linearly scaled to sum up to the observed sample size. Currently, `Ckmedian.1d.dp` only works with an equal weight of 1.
`method`	a character string to specify the speedup method to the original cubic runtime dynamic programming. The default is `"linear"`. All methods generate the same optimal results but differ in runtime or memory usage. See Details.
`estimate.k`	a character string to specify the method to estimate optimal `k`. This argument is effective only when a range for `k` is provided. The default is `"BIC"`. See Details.

Details

`Ckmean.1d.dp` minimizes unweighted or weighted within-cluster sum of squared distance (L2).

`Ckmedian.1d.dp` minimizes within-cluster sum of distance (L1). Only unweighted solution is implemented and guarantees optimality.

In contrast to the heuristic k-means algorithms implemented in function `kmeans`, this function optimally assigns elements in numeric vector `x` into `k` clusters by dynamic programming (Wang and Song, 2011). It minimizes the total of within-cluster sums of squared distances (*withinss*) between each element and its corresponding cluster mean. When a range is provided for `k`, the exact number of clusters is determined by Bayesian information criterion. Different from the heuristic k-means algorithms whose results may be non-optimal or change from run to run, the result of Ckmeans.1d.dp is guaranteed to be optimal and reproducible, and its advantage in efficiency and accuracy over heuristic k-means methods is most pronounced at large k.

The `estimate.k` argument specifies the method to select optimal `k` based on the Gaussian mixture model using the Bayesian information criterion (BIC). When `estimate.k="BIC"`, it effectively deals with variance estimation for a cluster with identical values. When `estimate.k="BIC 3.4.12"`, it uses the code in version 3.4.12 and earlier to estimate `k`.

The `method` argument specifies one of three options to speed up the original dynamic programming taking a runtime cubic in sample size n. The default `"linear"` option, giving a total runtime of $O(n \lg n + kn)$ or $O(kn)$ (if `x` is already sorted in ascending order) is the fastest option but uses the most memory (still $O(kn)$); the `"loglinear"` option, with a runtime of $O(kn \lg n)$, is slightly slower but uses the least memory; the slowest `"quadratic"` option, with a runtime of $O(kn^2)$, is provided for the purpose of testing on small data sets.

When the sample size n is too large to create two $k \times n$ dynamic programming matrices in memory, we recommend the heuristic solutions implemented in the `kmeans` function in package **stats**.

Value

An object of class "Ckmeans.1d.dp" or "Ckmedian.1d.dp". It is a list containing the following components:

cluster	a vector of clusters assigned to each element in x. Each cluster is indexed by an integer from 1 to k.
centers	a numeric vector of the (weighted) means for each cluster.
withinss	a numeric vector of the (weighted) within-cluster sum of squares for each cluster.
size	a vector of the (weighted) number of elements in each cluster.
totss	total sum of (weighted) squared distances between each element and the sample mean. This statistic is not dependent on the clustering result.
tot.withinss	total sum of (weighted) within-cluster squared distances between each element and its cluster mean. This statistic is minimized given the number of clusters.
betweenss	sum of (weighted) squared distances between each cluster mean and sample mean. This statistic is maximized given the number of clusters.
xname	a character string. The actual name of the x argument.
yname	a character string. The actual name of the y argument.

Each class has a print and a plot method, which are described along with print.Ckmeans.1d.dp and plot.Ckmeans.1d.dp.

Author(s)

Joe Song and Haizhou Wang

References

Wang H, Song M (2011). "Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming." *The R Journal*, **3**(2), 29–33. https://doi.org/10.32614/RJ-2011-015.

See Also

ahist, plot.Ckmeans.1d.dp, print.Ckmeans.1d.dp in this package.

kmeans in package **stats** that implements several heuristic k-means algorithms.

Examples

```
# Ex. 1 The number of clusters is provided.
# Generate data from a Gaussian mixture model of three components
x <- c(rnorm(50, sd=0.2), rnorm(50, mean=1, sd=0.3), rnorm(100,
       mean=-1, sd=0.25))
# Divide x into 3 clusters
k <- 3

result <- Ckmedian.1d.dp(x, k)

plot(result, main="Optimal univariate k-median given k")
```

```
result <- Ckmeans.1d.dp(x, k)

plot(result, main="Optimal univariate k-means given k")

plot(x, col=result$cluster, pch=result$cluster, cex=1.5,
     main="Optimal univariate k-means clustering given k",
     sub=paste("Number of clusters given:", k))
abline(h=result$centers, col=1:k, lty="dashed", lwd=2)
legend("bottomleft", paste("Cluster", 1:k), col=1:k, pch=1:k,
       cex=1.5, bty="n")

# Ex. 2 The number of clusters is determined by Bayesian
#       information criterion
# Generate data from a Gaussian mixture model of three components
x <- c(rnorm(50, mean=-3, sd=1), rnorm(50, mean=0, sd=.5),
       rnorm(50, mean=3, sd=1))
# Divide x into k clusters, k automatically selected (default: 1~9)

result <- Ckmedian.1d.dp(x)

plot(result, main="Optimal univariate k-median with k estimated")

result <- Ckmeans.1d.dp(x)

plot(result, main="Optimal univariate k-means with k estimated")

k <- max(result$cluster)
plot(x, col=result$cluster, pch=result$cluster, cex=1.5,
     main="Optimal univariate k-means clustering with k estimated",
     sub=paste("Number of clusters is estimated to be", k))
abline(h=result$centers, col=1:k, lty="dashed", lwd=2)
legend("topleft", paste("Cluster", 1:k), col=1:k, pch=1:k,
       cex=1.5, bty="n")

# Ex. 3 Segmenting a time course using optimal weighted
#       univariate clustering
n <- 160
t <- seq(0, 2*pi*2, length=n)
n1 <- 1:(n/2)
n2 <- (max(n1)+1):n
y1 <- abs(sin(1.5*t[n1]) + 0.1*rnorm(length(n1)))
y2 <- abs(sin(0.5*t[n2]) + 0.1*rnorm(length(n2)))
y <- c(y1, y2)

w <- y^8 # stress the peaks
res <- Ckmeans.1d.dp(t, k=c(1:10), w)
plot(res)
plot(t, w, main = "Time course weighted k-means",
     col=res$cluster, pch=res$cluster,
     xlab="Time t", ylab="Transformed intensity w",
     type="h")
abline(v=res$centers, col="chocolate", lty="dashed")
```

```
text(res$centers, max(w) * .95, cex=0.5, font=2,
     paste(round(res$size / sum(res$size) * 100), "/ 100"))
```

Univariate Segmentation

Optimal Univariate Segmentation

Description

Perform optimal univariate k-segmentation.

Usage

```
Cksegs.1d.dp(y, k=c(1,9), x=seq_along(y),
             method=c("quadratic", "linear", "loglinear"),
             estimate.k=c("BIC", "BIC 3.4.12"))
```

Arguments

y	a numeric vector of y values. Values can be negative.
k	either an exact integer number of clusters, or a vector of length two specifying the minimum and maximum numbers of clusters to be examined. The default is c(1,9). When k is a range, the actual number of clusters is determined by Bayesian information criterion.
x	an optional numeric vector of data to be clustered. All NA elements must be removed from x before calling this function. The function will run faster on sorted x (in non-decreasing order) than an unsorted input.
method	a character string to specify the speedup method to the original cubic runtime dynamic programming. The default is "quadratic", which generates optimal results. The other options do not guarantee optimal solution and differ in runtime or memory usage. See Details.
estimate.k	a character string to specify the method to estimate optimal k. The default is "BIC". See Details.

Details

Cksegs.1d.dp minimizes within-cluster sum of squared distance on y. It offers optimal piecewise constant approximation of y within clusters of x. Only method="quadratic" guarantees optimality. The "linear" and "loglinear" options are faster but not always optimal and are provided for comparison purposes.

The Bayesian information criterion (BIC) method to select optimal k is updated to deal with duplicates in the data. Otherwise, the estimated k would be the same with previous versions. Set estimate.k="BIC" to use the latest method; use estimate.k="BIC 3.4.12" to use the BIC method in version 3.4.12 or earlier to estimated k from the given range. This option is effective only when a range for k is provided.

`method` specifies one of three options to speed up the original dynamic programming taking a runtime cubic in sample size n. The default `"quadratic"` option, with a runtime of $O(kn^2)$, guarantees optimality. The next two options do not guarantee optimality. The `"linear"` option, giving a total runtime of $O(n \lg n + kn)$ or $O(kn)$ (if `x` is already sorted in ascending order) is the fastest option but uses the most memory (still $O(kn)$); the `"loglinear"` option, with a runtime of $O(kn \lg n)$, is slightly slower but uses the least memory.

Value

An object of class `"Cksegs.1d.dp"`. It is a list containing the following components:

`cluster`	a vector of clusters assigned to each element in `x`. Each cluster is indexed by an integer from 1 to `k`.
`centers`	a numeric vector of the (weighted) means for each cluster.
`withinss`	a numeric vector of the (weighted) within-cluster sum of squares for each cluster.
`size`	a vector of the (weighted) number of elements in each cluster.
`totss`	total sum of (weighted) squared distances between each element and the sample mean. This statistic is not dependent on the clustering result.
`tot.withinss`	total sum of (weighted) within-cluster squared distances between each element and its cluster mean. This statistic is minimized given the number of clusters.
`betweenss`	sum of (weighted) squared distances between each cluster mean and sample mean. This statistic is maximized given the number of clusters.
`xname`	a character string. The actual name of the `x` argument.
`yname`	a character string. The actual name of the `y` argument.

The class has a print and a plot method: `print.Cksegs.1d.dp` and `plot.Cksegs.1d.dp`.

Author(s)

Joe Song

See Also

`plot.Cksegs.1d.dp` and `print.Cksegs.1d.dp`.

Examples

```
# Ex 1. Segmenting by y

y <- c(1,1,1,2,2,2,4,4,4,4)

res <- Cksegs.1d.dp(y, k=c(1:10))

main <- "k-segs giving 3 clusters\nsucceeded in finding segments"

opar <- par(mfrow=c(1,2))

plot(res, main=main, xlab="x")
```

```
res <- Ckmeans.1d.dp(x=seq_along(y), k=c(1:10), y)
main <- "Weighted k-means giving 1 cluster\nfailed to find segments"

plot(res, main=main, xlab="x")

par(opar)

# Ex 2. Segmenting by y

y <- c(1,1,1.1,1, 2,2.5,2, 4,5,4,4)
res <- Cksegs.1d.dp(y, k=c(1:10))
plot(res, xlab="x")

# Ex 3. Segmenting a sinusoidal curve by y
x <- 1:125
y <- sin(x * .2)
res.q <- Cksegs.1d.dp(y, k=8, x=x)
plot(res.q, lwd=3, xlab="x")

# Ex 4. Segmenting by y

y <- rep(c(1,-3,4,-2), each=20)
y <- y + 0.5*rnorm(length(y))
k <- 1:10
res.q <- Cksegs.1d.dp(y, k=k, method="quadratic")
main <- paste("Cksegs (method=\"quadratic\"):\ntot.withinss =",
              format(res.q$tot.withinss, digits=4), "BIC =",
              format(res.q$BIC[length(res.q$size)], digits=4),
              "\nGUARANTEE TO BE OPTIMAL")
plot(res.q, main=main, xlab="x")
res.l <- Cksegs.1d.dp(y, k=k, method="linear")
main <- paste("Cksegs (method=\"linear\"):\ntot.withinss =",
               format(res.l$tot.withinss, digits=4), "BIC =",
              format(res.l$BIC[length(res.l$size)], digits=4),
               "\nFAST BUT MAY NOT BE OPTIMAL")
plot(res.l, main=main, xlab="x")
res.g <- Cksegs.1d.dp(y, k=k, method="loglinear")
main <- paste("Cksegs (method=\"loglinear\"):\ntot.withinss =",
              format(res.g$tot.withinss, digits=4), "BIC =",
              format(res.g$BIC[length(res.g$size)], digits=4),
              "\nFAST BUT MAY NOT BE OPTIMAL")
plot(res.g, main=main, xlab="x")
```

Index

∗Topic **cluster**
ahist, 4
Ckmeans.1d.dp-package, 2
MultiChannel.WUC, 7
plot.Ckmeans.1d.dp, 9
plot.Cksegs.1d.dp, 10
plotBIC, 11
print.Ckmeans.1d.dp, 12
print.Cksegs.1d.dp, 13
Univariate Clustering, 14
Univariate Segmentation, 18
∗Topic **distribution**
ahist, 4
plot.Ckmeans.1d.dp, 9
plot.Cksegs.1d.dp, 10
∗Topic **hplot**
ahist, 4
plot.Ckmeans.1d.dp, 9
plot.Cksegs.1d.dp, 10
plotBIC, 11
∗Topic **package**
Ckmeans.1d.dp-package, 2
∗Topic **print**
print.Ckmeans.1d.dp, 12
print.Cksegs.1d.dp, 13
∗Topic **univar**
ahist, 4
Ckmeans.1d.dp-package, 2
MultiChannel.WUC, 7
plot.Ckmeans.1d.dp, 9
plot.Cksegs.1d.dp, 10
plotBIC, 11
print.Ckmeans.1d.dp, 12
print.Cksegs.1d.dp, 13
Univariate Clustering, 14
Univariate Segmentation, 18

ahist, *3*, 4, *16*

Ckmeans.1d.dp, *3*, *5*, *9*, *12*, *13*
Ckmeans.1d.dp *(*Univariate Clustering*)*, 14
Ckmeans.1d.dp-package, 2
Ckmedian.1d.dp, *9*
Ckmedian.1d.dp *(*Univariate Clustering*)*, 14
Cksegs.1d.dp, *10*, *11*, *14*
Cksegs.1d.dp *(*Univariate Segmentation*)*, 18

graphics, *4*, *6*

hist, *4–6*

kmeans, *4*, *15*, *16*

MultiChannel.WUC, 7

plot, *9*, *10*, *12*
plot.Ckmeans.1d.dp, *3*, 9, *16*
plot.Ckmedian.1d.dp *(*plot.Ckmeans.1d.dp*)*, 9
plot.Cksegs.1d.dp, 10, *19*
plot.histogram, *5*
plotBIC, *3*, 11
print, *13*, *14*
print.Ckmeans.1d.dp, *3*, 12, *16*
print.Ckmedian.1d.dp *(*print.Ckmeans.1d.dp*)*, 12
print.Cksegs.1d.dp, 13, *19*

stats, *4*, *15*, *16*

Univariate Clustering, 14
Univariate Segmentation, 18